(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 885 424 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
02.02.2000 Bulletin 2000/05

(21) Application number: **97904603.4**

(22) Date of filing: **21.02.1997**

(51) Int Cl.[7]: **G06F 17/30**

(86) International application number:
PCT/JP97/00494

(87) International publication number:
WO 97/33239 (12.09.1997 Gazette 1997/39)

(54) DATABASE SYSTEMS HAVING SINGLE-ASSOCIATION STRUCTURES AND METHOD FOR SEARCHING DATA IN THE DATABASE SYSTEMS

DATENBANKSYSTEM MIT EINZELASSOZIATIONSSTRUKTUREN UND VERFAHREN ZUM SUCHEN VON DATEN IN DATENBANKSYSTEMEN

SYSTEMES DE BASES DE DONNEES A STRUCTURES D'ASSOCIATIONS UNIQUES ET PROCEDE DE RECHERCHE DE DONNEES DANS LES SYSTEMES DE BASES DE DONNEES

(84) Designated Contracting States:
DE FR GB

(30) Priority: **05.03.1996 US 611293**

(43) Date of publication of application:
23.12.1998 Bulletin 1998/52

(73) Proprietor: **Sofmap Future Design Co., Ltd.**
Tokyo 101 (JP)

(72) Inventors:
- **TABUCHI, Daisuke**
  Sotokanda, Chiyoda-ku, Tokyo 101 (JP)
- **SHOJI, Wataru**
  Sotokanda, Chiyoda-ku, Tokyo 101 (JP)
- **NAKAJIMA, Ichiro**
  Sotokanda, Chiyoda-ku, Tokyo 101 (JP)

(74) Representative: **Heusler, Wolfgang et al**
v. Bezold & Sozien,
Patentanwälte,
Akademiestrasse 7
80799 München (DE)

(56) References cited:
EP-A- 0 550 368
EP-A- 0 690 398
WO-A-90/08360

EP 0 885 424 B1

Printed by Jouve, 75001 PARIS (FR)

Description

FIELD OF THE INVENTION

[0001] The present invention relates to a database system, and more particularly relates to database system in which data is organized in a plurality of single-association structure.

BACKGROUND OF THE INVENTION

[0002] Databases are among the most important assets of a company. Examples of databases are customer records (e.g., name, address, phone number, credit card number, etc.), accounting data (e.g., account receivable and account payable), and mailing lists (for soliciting new customers). A company would plunge into financial crisis if its databases are destroyed. For example, the company will not be able to collect its bills if its account receivable files are erased. Thus, the value of the databases of a company could be higher than the value of its tangible assets.

[0003] A database is a collection of data organized in a predetermined manner. Typically, the data is arranged into a plurality of records having identical structure. Each record contains one or more fields (e.g., name, address, and phone numbers) for holding appropriate data. In order to retrieve data quickly, a database is typically associated with an index which contains a table of keys representing every record in the database. The keys typically correspond to the data in a predetermined field (e.g., name) of the records. The keys are arranged in a user-defined criterion (e.g., ascending or descending alphabetical order). Generally, records are stored in the database in the order they are entered. In most cases, records are entered without any particular order, and can be considered random. However, the index is rearranged in accordance with the criterion every time when one or a set of records are entered. When it is desirable to locate a record, the index is searched first. Because the index is well-organized, the search time is very short. The correct record can be easily found based on the information obtained from the search result. If the index in not used, the entire database (which could be many times the size of the index) needs to be searched.

[0004] The sizes of these databases are typically very big (several million bytes). The number of rows and columns are very large. There could be many indexes associated with a database. Further, the search "engines" are very complex because they have to handle the huge databases and work with all the indexes. As a result, even in the age of personal computers, these databases are best handled by mainframe computers. Patent application WO 90/08360 already discloses such database systems whereby these systems can even include several heterogeneous databases.

[0005] There are "free-form" databases in which the data is not organized in any predetermined manner while allowing every word in the database to be searched. This type of databases are often used by online information providers which provide full-text searching capability to newspapers, magazines, court decisions, etc. In this type of databases, all the words in the text are indexed. The "free-form" databases allow for efficient retrieval of data, but is typically not flexible and very inefficient in utilizing computer resources (e.g., the index files are typically very large).

[0006] Thus, there is a need to design a simple, fast, flexible and efficient database system.

SUMMARY OF THE INVENTION

[0007] The present invention involves a novel database system comprising a plurality of single-association databases each associated with a database driver, and a method for searching data in the database system. A single-association database contains a plurality of records, each record associates one piece of data with another piece of data. One implementation of the single-association database is an ASCII file containing many lines, each line associate one set of ASCII characters with another set of ASCII characters. An example of such a file is:

A111 = John

A113 = Peter

B111 = Mary

[0008] In this example, the symbol "=" is an association symbol, and the set of ASCII characters to the left of the "=" sign (e.g., "A111") is associated with the set of characters to the right of the "=" sign (e.g., "John").

[0009] The database driver is a software routine optimized for searching information in its associated database. As a result, the size of the driver is small and the search speed is fast. Further, the plurality of single-association databases can be combined in a variety of ways so that complicated searches can be easily conducted. These combinations give rise to a very flexible database system.

[0010] The present database system does not need to have index files. The structure of each database in the present system is often simpler than an index file of a conventional database system. As a result, there is no need to construct an index file to speed up searching. The lack of index files is another advantage of the present database system.

[0011] In this database system, all the database drivers and their associated databases are hierarchically equal. Architecturally, these drivers and databases are independent of each other. Adding more databases will not increase the complexity of the system, although the information contained in the database system increases. As a result, the size of the present database system

can be increased without penalizing its performance.

[0012] The database system of the present invention can be easily implemented on a network (both local and wide area). The databases and the drivers could reside on different computers of a network. A computer can combine its existing databases and drivers with the databases and drivers downloaded from other computers so as to build a desired database system comprising a plurality of single-association databases.

[0013] The database system of the present invention can be implemented using a novel bossless computer program architecture (called "digital cell technology") comprising a plurality of program modules called "cells. " Under this architecture, each cell is hierarchically equal, i.e., there is no controlling (or boss) cell. An application can start from any cell, and can terminate at any cell. Typically, many cells are executing either sequentially or concurrently. Various applications can be designed by controlling the operation of these cells. As explained above, the database system of the present invention comprises a plurality of database drivers which are hierarchically equal. This structure is compatible with the digital cell technology. In this embodiment, the database and interface drivers could be implemented as cells.

[0014] Each cell is associated with a file, called a DNA file. The characteristics and operation of the cells are determined by their associated DNA files. Cells communicate by writing statements to the DNA files associated with other cells using a protocol called digital shifting function (DSF). Once written, the origin of these DSF statements is ignored. There is no need to "return" to the cells which originate the statements. Further, the DSF statements are executed by the cells without regard to their origins.

[0015] The cells execute statements in their associated DNA files. These statements are executed sequentially. The cells retain full control of the execution, i.e., there is no need to turn over execution to other cells during or after the execution of statements. There is no need to report to other cells on the status or results of execution.

[0016] The digital cell technology does not require the preservation of linkage information because there is no need for the cells to return information or control to other cells. This technology is flexible because each cell is hierarchically at the same level as other cells, thus can be selected to perform a certain task based only on its ability instead of on its hierarchic level. For example, when there is no need for a cell to exist, it can be removed from an application. On the other hand, program modules in conventional boss architecture cannot be removed arbitrarily. For example, the main program cannot be removed, even if it is not performing any useful function.

[0017] In one embodiment of the database system, the single-association databases could be embedded in the DNA files of the associated database drivers. In this way, the database drivers are integrated with their associated databases. Some of the statements executable by the database drivers relate to searching of their associated databases. Thus, other cells can request a database driver to search its associated database by sending a search statement to the DNA file of the database driver. The requesting cells can also specify the destination of the search result. The destination could be the requesting cell's own DNA file, the driver cell's DNA file, or an independent file. Thus, the present database system can be easily designed to interact with non-database related cells. This level of flexibility allows more efficient and powerful database systems to be designed.

[0018] In applications formed by a plurality of cells (including database drivers, requesting cells and other cells), each cell is at the same level as the other cells. Thus, each database driver and requesting cell can execute its designed functions independent of each other and can interact with any other cells it desires. Each cell can issue statements to another cells without going through a chain of cells. The mode of interaction of the present invention is direct (i.e., from one cell to another directly instead of through a chain of modules). As a result, the structure of an application is simple and the execution speed is fast.

[0019] These and other features and advantages can be understood from the following detailed description of the invention together with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] Fig. 1 is a schematic diagram of a database system of the present invention.

[0021] Fig. 2 is an exemplary graphic display of a database driver of the present invention.

[0022] Fig. 3 is an exemplary graphic display of an interface driver of the present invention.

[0023] Fig. 4 is a schematic diagram of the database system of the present invention implemented on a network system.

[0024] Fig. 5 is a schematic diagram of an application based on a boss less architecture which can be used to implement the drivers of the present invention.

[0025] Fig. 6 is a drawing showing the structure of an application using a digital cell technology.

[0026] Fig. 7 is a block diagram showing the logical structure of a DNA file associated with a cell CA

[0027] Fig. 8 is a block diagram of the logical structure of the cell CA.

[0028] Fig. 9 is a block diagram showing the logical structure of a DNA file associated with a database driver cell CB.

[0029] Fig. 10 is a drawing showing a flow of DSF statements when results of searching databases are combined.

[0030] Fig. 11 is a block diagram of a computer system which can be used to run the database system of

the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0031] The present invention is directed to a novel database system. The following description is presented to enable any person skilled in the art to make and use the invention. Descriptions of specific applications are provided only as examples. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[0032] Fig. 1 is a schematic drawing of a database system 700 of the present invention. Database system 700 contains a plurality of single-association databases, such as databases 704-706. One way to picture a single-association database is a table having many rows and two columns. The two columns in each row associate one set of data with another set of data. The data could be text, numbers, graphic, audio and video data. In an example involving a library database system used to keep track of all books in a library, the first column of database 704 could be the identification (ID) number of all books in the library while the second column could be the year of publication of each book. Thus, database 704 shows a single association between a pair of data. This is different from conventional database wherein each record contains many fields, thereby associating the data in one field with the data in many fields.

[0033] In order to be useful, many single-association databases are needed in system 700. Thus, another database (e.g., database 705) may associate the ID with the authors of the books while a third database (e.g., database 706) may associate the ID to the names of the persons who are borrowing the books. It is possible for the first column of the databases in system 700 to be different. For example, the first column of one of the databases in the above example may be the title of the books instead of the ID number. However, if it is desirable to link this database to the other databases, there is a need to have a database which link the ID to the title of the books.

[0034] Each database is associated with a database driver. The main function of the database driver is to perform searches on its associated database and return the results of the searches to an appropriate file. The database driver could also perform other functions on its associated database, such as editing the records in the database. In Fig. 1, databases 704-706 are associated with drivers 712-714, respectively.

[0035] In a conventional database system, a single driver (i.e., search routine) is associated with all the complex databases in the system. Thus, if a database system contains ten separate databases, the same driver would be used to perform search on the ten databases. Because the driver needs to handle all the requirements of a large number of complex databases, the size of the driver becomes very large. This is different from the database system of the present invention in which each driver is specially designed to operate on its associated database.

[0036] Fig. 2 shows an exemplary graphic display 740 optionally associated with database driver 712. The underlying database 704 provides the relationship between the ID number and the year of publication of books in the above-mentioned library. Display 740 contains a window 742 allowing a user to enter a relationship. In Fig. 2, the relationships are greater than (>), less than (<), equal to (=), greater than or equal to (>=), and less than or equal to (<=). It should be appreciated that other relationship can be included. Display 740 also contains a window 744 showing the set of possible data in the column (in this case, all the years within a range). Window 744 contains a scroll bar 745 allowing the user to scroll through all the available years. The user can define a search criterion by clicking on a relationship (e.g., greater than) in window 742 and a year (e.g., 1960) in window 744. If it is desirable to define a range, display 740 provides an "AND" button 746 for defining an AND relationship. Thus, the user can click on AND button 746, and then select another relationship (e.g., less than) in window 742 and a year (e.g., 1980) in window 744. The user can then click on a SEARCH button 748 to execute the search. The result of the search is display on a window 750. Window 750 contains a scroll bar 751 allowing the user to review all the books which meet the selection criterion.

[0037] To make the display more user friendly, display 700 may contain a window 752 showing the search criterion. Display 740 may contain other buttons, such as a CANCEL button 754 or a button for forming an "OR" operation (or other logic variations) of the search relationships. For example, if the user wishes to search for books which are published in 1960 and 1980, the user can define the search relationship (using the above described windows) and then click on the "OR" button.

[0038] Typically, an application needs to perform a search on more than one databases. For example, the application may like to search for all books published by a certain publisher between 1960 and 1970. This search would involve (i) a search on a database associating the ID with the publishers and (ii) a search on a database associating the ID with the years of publication. The results of the two searches are then "ANDed" together.

[0039] Returning now to Fig. 1, an interface driver 720 is used to manage searches on multiple databases. One of the functions of an interface routine 720 is to allow an application 724 to specify the databases to be searched and the search criteria.

[0040] Fig. 3 shows an exemplary display 770 optionally associated with interface driver 720. Display 770

contains a plurality of windows (such as windows 772-775), one for displaying information relating to a selected database. Each one of windows 772-775 has a similar structure as display 740 of Fig. 2, and consequently will not be shown in detail. The user can select search criteria in each window using the methods described above. The result of the search for each database is displayed in the corresponding window. The user can then click on one of the logic relationships shown in a window 778 to select a search for the results of all the databases. A click on a SEARCH button 782 starts the search. The final result is displayed on a result window 780. For example, if the "AND" row in window 778 is selected, window 780 would show the AND of the search results in windows 772-775.

[0041] Display 770 may contain other buttons, such as a CANCEL button 784.

[0042] The present invention is different from prior art database systems which contain a small number of multi-association databases and a search routine. Typically, the structures of the prior art databases are very complex because each database is designed to contain as much information as possible. As a result, the search routine is also very complex because it has to understand the complicated structure of the underlying database. Consequently, the databases and search routine are very difficult to use and maintain.

[0043] The database system of the present invention is especially useful in a distributed environment over a local or wide area network. These networks comprise a plurality of computers. Some of the computers can be used to develop databases and associated drivers of the present invention. Other computers can download and use these databases. An example of a wide-area network is the Internet.

[0044] Fig. 4 is a schematic diagram showing a network system 800 of the present invention. System 800 comprises a plurality of computers 802, 804,806, 808 and 810. These computers are coupled to a network 812, which could be a local or wide area network. A database (and its associated driver) 820 has been developed by and is stored in computer 802. Two databases 824 and 825 reside in computer 804. Computer 804 can download database 820 from computer 802 using a conventional network protocol. The downloaded database 820A is shown in Fig. 4 as dashed lines. Applications in computer 804 can now use all three databases, including 820A downloaded from computer 802. Similarly, computer 808 can download database 820 from computer 802 using a conventional network protocol. The downloaded database 820B is shown in Fig. 4 as dashed lines. Applications in computer 808 can now use an existing database 828 and database 820B downloaded from computer 802.

[0045] It should be noted that a database and its associated driver could reside on different computers on the network. For example, a database could reside on computer 806 while its associated driver could reside on computer 810. A computer wishing to use the database needs to download the database and the associated driver from computers 806 and 810.

[0046] Because the structure of the databases and drivers of the present invention could be very simple, even low performance personal computers and relatively unskilled users could design valuable databases. These databases and drivers can be easily downloaded and used by other computers in the network. Further, only those databases and drivers that are actually used by a computer need to be downloaded, thereby reducing traffic on the network. On the other hand, designing conventional databases is a complex task that can be handled only by experienced database designers using powerful computers. Transferring a conventional database on a network requires considerable bandwidth because the size of a conventional database is typically very large.

[0047] The database system of the present invention contains single-association databases and related drivers. The databases are very flexible, and can be changed and added easily. The drivers are designed to work with their associated database. Because the underlying databases have a simple structure, the drivers also have a simple structure and cam be designed very efficiently.

[0048] One aspect of the database system of the present invention is to reduce the underlying structure to a simple form. This aspect is compatible with a new computer software architecture called "digital cell technology." As a result, the database of the present invention can best be implemented using the digital cell technology. A detailed description of this technology has been disclosed in copending the United States patent application (Serial No. 08/539,806) filed October 5, 1995 and corresponding International patent application (Serial No. JP 96/00821) filed March 28, 1996. These copending patent applications are hereby incorporated by reference.

[0049] An implementation of the present inventive database system using digital cell technology is now described. One characteristic of the digital cell technology is that it is a "bossless" architecture because every program module (or cell) is on equal footing with other program modules. There is no module that controls the overall operation of the program (i.e., no boss).

[0050] Fig. 5 is a schematic view of an application 160 based on the "bossless" digital cell technology. Application 160 comprises a plurality of program modules, such as modules 162-165. Each program module (called a "cell") is the same as the other cells in an hierarchical sense. Cells are linked together in a way in which no history or linkage information needs to be retained. Each link is independent. For example, there is no need for links to be active simultaneously. Each link is direct, i. e., two cells can be linked directly without the need of using one or more intermediate links. For example, cells 162 and 164 can be linked directly using line 166 instead

of using lines 167 and 168 and passing through an intermediate cell. An application can be formed by defining the cells involved and using the direct link.

[0051] Fig. 6 is a drawing showing the structure of an application 200 using the bossless architecture of the present invention. Application 200 contains a plurality of cells, labeled as C1-C5, loaded and executing in RAM. Each cell has an associated file (labeled as D1-D5), called DNA file, which contains information of the cell. The term "DNA" is used here in analogy with the biological relationship between a living cell and its DNA. At a desired time, cell C1 can send statements (called "DSF statements") to cell C2 using a protocol called digital shifting function ("DSF") protocol. Cell C2 will execute these statements. The detail structures of cells, DNA files and the DSF protocol will be described below.

[0052] In this application 200, Cell C2 does not retain information on the origin of these statements, i.e., no history of the inter-process communication is kept. Thus, once cell C1 completes its writing of DSF statements to cell C2, there is no further linkage between cells C1 and C2. Cell C2 does not know the origin of these statements during their execution. It is possible for cell C1 to later establish communication with cell C2 again by sending another set of statements to C2. However, this communication is separate from the previous communication, and terminates once the new set of DSF statements is sent.

[0053] Each of the cells can send DSF statements to any of the cells it desires. Thus, cell C1 can also send statements to cell C3. Similarly, cell C2 can send statements to cell C4, which in turn can send statements to cell C1. Cell C3 can also send statements to cell C1.

[0054] In this example, cells C1 and C2 are not bosses to C4. For example, when C4 is executing DSF statements, there is no need to maintain any links between cells C1 and C2 and between cells C2 and C4. Cell C4 has no obligation to report results of execution to any cells in application 200. Links are maintained only during the time DSF statements are transferred. Further, the writing of statements by cell C1 to cell C2 could be independent of the writing of statements by cell C2 to cell C4. In addition, cell C4 merely execute statements, and does not care where the statements come from.

[0055] In this architecture, there is no need to save and restore register values on a stack when cell C2 executes statements written by cell C1. There is no need to register cells in a central database prior to sending commands. There is no need to send messages back and forth to report status of execution. As a result, the application can be executed quickly.

[0056] As explained below, a cell can be implemented as an "EXE" file (in the MSDOS or MS Windows environment) and can be invoked by loading into RAM for execution using well known methods in accordance with the operating environment. The cell's associated DNA file can also be loaded into RAM. The invoked cell takes on the attributes stored in its DNA file. It is also possible to modify the DNA file when the cell is invoked or while running by writing to the file (which could be an ASCII file).

[0057] Fig. 7 is a block diagram showing the logic structure of a DNA file 250 associated with a cell, such as cell CA. File 250 has a section 252 containing parameters ("own parameters") related to the characteristics of cell CA itself. For example, section 252 may contain parameters related to the way cell CA manifest itself when invoked: the window size and background color of cell CA, the name of cell CA, the names of audio files associated with its invocation and termination, etc.

[0058] File 250 also contains a section 254 containing linking parameters ("link parameters") on cells related to cell CA. Examples of the parameters contained in this section are: the names, symbols and positions of the other cells. One of the parameter is "close," which is interpreted as closing cell CA when the cell associated with this parameter is invoked.

[0059] File 250 further contains a DSF information section 256. This section contains a regular section 257 and a top priority function section 264. The structure of the regular section 257 and top priority function section 264 are substantially the same, except that top priority function section 264 has a higher priority in DSF statement execution. These two sections contain individual headers for identifying the sections (e.g., each section headed by a different name or symbol).

[0060] Regular section 257 contains a "condition" section 258 and a statements section 260. Statements section 260 comprises statements sent to cell CA by other cells. Statements in statements section 260 are executed sequentially. Each statement also contains parameters necessary for the execution of the statement. Condition section 258 comprises three components: (a) a first pointer to the last DSF statement currently existing in statements section 260, (ii) a second pointer to the current DSF statement being processed by cell CA, and (iii) the current status of the cell.

[0061] Top priority function section 264 contains a condition section 266 and a command lines section 268. The structure of condition section 266 is similar to the structure of condition section 258. Command lines section 268 contains executable command lines which are sent by other cells using DSF (or similar) protocol. The command lines have a higher execution priority than the statements in statements section 260. The command lines in command lines section 268 are executed sequentially.

[0062] It should be appreciated that the logic structure shown in Fig. 7 can be implemented using one or more physical files. Further, portions of the logical sections may intermingle physically. In one embodiment, DNA file is a text file. Thus, the content of the DNA file can be modified by using a regular text editor.

[0063] Statements sent by one cell to another follow the DSF protocol. A sending cell (e.g., cell CS) sets up a communication link with the DNA file 250 associated

with cell CA. Specifically, it looks up the address of DNA file 250 and determines whether DNA file 250 is able to accept DSF statements by examining its status in condition section 258. Statements will be issued by cell CS only when cell CA is ready to accept them. In one embodiment, the issuance of statements involves writing ASCII characters to statements section 260 of DNA file 250.

[0064] When cell CS is authorized to issue statements to cell CA, cell CS reads the first pointer (in condition section 258) to the last DSF statement to determine the appropriate address to write the DSF statements. It is important not to overwrite DSF statements already existed in cell CA. Cell CS writes DSF statements to statements section 260 of DNA file 250. Cell CS also updates the first pointer in condition section 258 so as to point to the last DSF statement newly written into statements section 260. The communication link between cells CA and CS is terminated. It can be seen that cell CA and DNA file 250 do not maintain record (i.e., history) indicated that these new statements originate from cell CS.

[0065] It should be appreciated that the above described DSF protocol is only an exemplary protocol. Other protocol could be used to write DSF statements to cells. For example, different pointer structures can be used, e.g., the first pointer can point to the position after the last DSF statement. Different types of status and different ways for checking status information can be used. Further, the statements could be stored in accordance with a logic structure instead of stored physically in a sequential manner. For example, the statements could be organized into groups with the address of each group pointed to by a pointer.

[0066] Command lines are sent by one cell to another using a protocol substantially the same as the DSF protocol. Because regular statements section 257 and top priority function section 264 have different headers, the sending cell can distinguish between these two sections and write to the appropriate section. Other means for identifying these two section can also be used.

[0067] Fig. 8 shows the structure of cell CA. It is grouped logically into a plurality of sections, each implemented using instructions executable by a computer. Cell CA contains an initialization section 312 and a DNA interface section 314. DNA interface section 314 allows cell CA to read from and write to its corresponding DNA file 250. Initialization section 312 takes care of housekeeping tasks when invoked, including reading parameters from "own parameters" section 252 of DNA file 250. Cell CA also contains a DSF interface section 316 (for processing DSF protocol). This section contains code (or program instructions) for sending and receiving statements/command lines using the DSF protocol.

[0068] Cell CA contains an execution section 318 containing code for automatically executing statements and command lines in DNA file 250 written by other cells. The code sequentially read and execute statements in statements section 260 of DNA file 250. After each statement is executed, cell CA branches to top priority function section 259 and executes all the command lines therein. Cell CA then executes the next statement in statement section 260.

[0069] Cell CA contains a temporary memory section 322 for storing temporary information. As an example, it is possible to change attributes (e.g., background color and the size of the display window) of cell CA during its execution. In one embodiment, the changed attributes are temporarily stored in temporary memory section 322 instead of immediately being written to DNA file 250. In this embodiment of cell CA, the attribute information stored in temporary memory section 322 is written into "own parameters" section 252 of DNA file 250 only when cell CA is terminated.

[0070] Cell CA also contains a cell invocation section 324 for invoking other cells. In one embodiment, this section obtains information about the cell desired to be invoked and pass this information to a specialized cell which actually invoke the desired cell. It is also possible to incorporate the functionality of this specialized cell in the cell invocation section 324 of cell CA and other cells.

[0071] It should be appreciated that the above mentioned sections in cell CA are prouped logically, and portions of these sections could intermingle physically.

[0072] It can be seen from the above described structures of cell CA and its associated DNA file 250 that both cell CA and DNA file 250 do not keep track of the origin of the DSF statements. A cell may accept DSF statements (stored in its associated DNA file) from many cells. After the DSF statements have been received, the linkage between the originating and destination cells is terminated. The cell executes DSF statements contained in its associated DNA file without knowledge of how the statements arrive the DNA file. As a result, there is no need to "return" to any cell. As pointed out above in connection with the destination file, nothing in this architecture prevents the recipient cell from returning information to the originating cell.

[0073] Typically, the size of each cell is small and the function of the cell well defined. Consequently, the execution speed is fast. As a result of the small size and specialized function, the cells can be easily written to fully utilize the resources of a computer. The communication between cells using DSF is direct, with minimum amount of access to the operating system on which an application is run. As a result, the efficiency is high.

[0074] The architecture of the digital cell technology comprises at least two cells which can communicate with each other. The cells are encapsulated program modules that are specialized for their designed tasks. Therefore, applications developed using this architecture comprise of multiple executable which can run independently or concurrently. The cells interact with each other using the inventive DSF protocol. Each cell can control the action of other cells. For example, a first cell can control a second cell, and the second cell can control the first cell. Therefore, no single cell has complete

control over the other cells, or in other words, there is no boss. Another unique characteristic of this architecture is that the cell that receives a command does not keep any information of where the command came from. This lack of historical knowledge allows cells to move forward instead of going backward in a link.

[0075] The above described feature of the digital cell technology is especially advantageous in implementing the database system of the present invention. As explained above, the present database system contains a plurality of database drivers designed to serve their associated database. These drivers are implemented as cells. Because there is no overhead in linking, the database system can support a large number of drivers without affecting the performance of the system. As a result, the digital cell technology provides an efficient platform to support the database the database system of the present invention. These two inventive technologies (digital cell and single-association database technologies) form a synergical cal combination that can outperform conventional database.

[0076] In the database system of the present invention, database drivers 712-714 and interface driver 720 can be implemented as cells. These cells can interact with other cell in the system in the manner described above. For example, database drivers 712-714 can receive "search" DSF statements from other cells for initiating a search on their associated database.

[0077] The digital cell technology is especially suitable for a networked database environment described above in Fig 4. The driver cells and the associated databases can be obtained from any source, including a remote computer in a network. Once downloaded to a local computer, they can interact with other cells in the local computer. The downloaded cells behave just like local cells.

[0078] It should be pointed out that the same database driver cell can run at different times, and access the same or different database at each time. For example, cell A can access database number one at time one and access database number two at time two, and then return to database number one at time three.

[0079] In one embodiment of the invention, the database of a database driver cell is stored in its associated DNA file. If cell CB is a database driver, file 250 contains a section 262 containing the associated database data.

[0080] Fig. 9 is a diagram showing the logical structure of DNA file 250 associated with driver cell CB. DNA file 250 comprises database section 262 having a database associated with driver cell CB. If the size of the database is large, the size of the database section also become large. For example, the contents of database section 262 is represented as shown below:

AC102 = John Smith

AC103 = Steve Dole

AC105 = Mike King

In this example, the symbol "=" is used for specifying association. The data to the left of the "=" sign (i.e., AC102, AC103 and AC105) represent ID numbers of books. On the other hand, the data to the right of the "=" sign represent author's name. These data could be the set of ASCII characters. The number of lines is determined in accordance with the amount of information in the database.

[0081] It is possible that database associated with cell CB be stored in other locations. In this case, file 250 contains a line which indicates the location of the associated database file.

[0082] If each of single-association databases 704 to 706 for library database system described with reference to Fig. 1 is accomplished as DNA file 250, data (ID and published year, ID and author's name, and ID and borrower's name) of each of single-association databases 704 to 706 are described on database section 262 of each DNA file.

[0083] Top priority function section 264 and regular section 257 of driver cell CB mainly receive DSF statements which perform instructions such as searching database and deleting the record in the database.

[0084] These DSF statements may comprise various parameters such as a search criteria parameter and a destination file parameter (for indicating the location of the file which used to store the results of the search). DSF information section 256 may include search key section 271 (for accepting search criteria) and destination file section 273. Other cells can write to these two sections as a part of a search request. The destination file could be the DNA file of the cell which requests the search, or the DNA file of the current database driver, or another file.

[0085] For example, two statement, "search for ID linked to author John Smith: destination is cell CD" and " search for ID linked to author Mike King: destination is cell CD," exist in statement section 260, and command line "search for ID linked to author Steve Dole: destination is cell CE" is written on command line section 268 after starting of execution of statement "search for ID linked to author John Smith: destination is cell CD."

[0086] Cell CB reads statement "author searches for ID of John Smith: destination is cell CD" of statement section 260, then searches the database in database section 262 for the record including "John Smith." When cell CB detects associated record, cell CB reads ID "AC102" of the record and sends "AC102" to cell CD (not shown) indicated by destination file parameter.

[0087] Cell CB looks command line section 268. Because command line "search for ID linked to author Steve Dole: destination is cell CE" exists in command line section 268, cell CB searches the database for the record including "Steve Dole." When cell CB detects associated record, cell CB reads ID "AC103" and sends it to cell CE (not shown) indicated by destination file pa-

rameter.

[0088] Then, cell CB goes back to statement section 260 and executes next statement "search for ID linked to author Mike King: destination is cell CD." When cell CB detects the record including "Mike King" in the database, cell CB reads ID "AC105" and sends it to cell CD indicated by destination file parameter.

[0089] In this embodiment, if command line in command line section 268 is "delete the record linked to author Mike King," cell CB deletes the record "AC105 = Mike King" in database section 262. In this case, If statement "search for ID linked to author Mike King: destination is cell CB" is executed, the message saying "There is no associated data" is sent to cell CB.

[0090] The search results of a plurality of databases can be combined by using an interface cell. The flow of DSF statement in this case now will be described with reference to Fig. 10.

[0091] For example, driver cell E1 comprising data including ID and author's name of a book and driver cell E2 comprising data including ID and published year of a book exist. And interface cell G1 which combines the search results of these driver cells E1 and E2 exists.

[0092] When you search for ID of a book whose author's name is "Mike King" or for ID of a book whose published year is "1970", interface cell G1 sends DSF statement "search for ID linked to author Mike King: destination is cell G1" to DNA file 250 in driver cell EI (L1), and sends DSF statement "search for ID linked to the published year 1970" to DNA file 250 in driver cell E2 (L2).

[0093] DSF statement sent from interface cell G1 are written on each DNA file 250 of cell EI and cell E2, and executed. Cell EI and cell E2 send the search results to DNA file 250 of interface cell G1 (L3, L4). When the search results are written on DNA file 250 of interface cell G1, interface cell G1 executes the OR processing to these search results and obtains the final search results (that is, ID of a book whose author's name is "Mike King" and ID of a book whose published year is "1970").

[0094] In case of searching for ID of a book whose author's name is "Mike King," and besides, whose published year is "1970," interface cell G1 executes the AND processing to the search results received from each of driver cells EI and E2.

[0095] It can be seen from above that the digital cell technology is especially suitable for implementing the database system of the present invention. This database system comprises many database drivers, only few of them may be used at a time. Under the digital cell technology, all the drivers are implemented independently (instead of subroutines, as is the case in conventional architecture). Only those drivers that are actually being used are active. As a result, there is little, if any, reduction in performance by adding more drivers (and their associated database) to the system.

[0096] One of the embodiments of the present invention is an application development system which runs under Microsoft's MS Windows. In this environment, cells are programs stored as ".EXE" files and typically shows a window on a computer monitor when invoked. By linking these cells, a users can construct an application software just like assemble blocks. Each cell, with its specific function, is given another function or value through DSF protocol with other cells to produce a variety of applications.

[0097] Fig. 11 shows a block diagram of a computer system 600 which can be used to run the database system of the present invention. Computer system 600 comprises a computer 602 having a central processing unit (CPU) 604 and system memory 606, which could be random access memory (RAM) or read-only memory (ROM), coupled to a system bus 608. Computer 602 also contains a peripheral bus controller 612 for controlling a peripheral bus 614. Depending on the architecture of computer 602, peripheral bus 614 could be a PCI bus, VESI local bus, ISA bus, or other similar buses. Peripheral bus 614 allows peripheral boards to be connected to computer 602. Examples of peripheral boards are a video board 616, a serial board 620 and a data transfer board 622. CPU 604 and RAM 606 can communicate with the peripheral boards through peripheral bus controller 612.

[0098] Serial board 620 allows serial communication between computer 602 and one or more external serial devices, such as a mouse 636.

[0099] Video board 616 contains circuits to control a monitor 630 and display images thereon. Video board 616 also contains memory (not shown) associated with such display. The memory is preferably a special kind of memory integrated circuit device, called video RAM (VRAM), designed for video applications. The circuits draws images on monitor 630 based on information stored in the memory. The images are on monitor 630 are updated at predetermined time intervals. An example of an image is display 740 shown in Fig. 2A.

[0100] If computer system 600 is used to run programs in a windows-based environment, one or more windows, such as a windows 632 and 634, could be displayed on monitor 630.

[0101] Disk controller board 622 is connected to a hard disk 638 and a floppy disk driver 638. The MS Windows and the operating system are generally stored in hard disk 638. The cells can be stored in floppy diskettes and downloaded into hard disk 638. In one embodiment of the present invention, individual database driver and its associated database can be designed and stored in diskette. These diskettes can be distributed to end users. The diskettes can be loaded into hard disk 638. If other cells want to use one or more of these databases, the cells can invoke the corresponding drivers (i.e., cells). Alternatively, these new drivers can also invoke other cells.

[0102] The invention has been described with reference to a specific exemplary embodiment thereof. Various modification and changes may be made thereunto

without departing from the scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense; the invention is limited only by the provided claims.

Claims

1. A database system comprising:

   at least a first and a second single-association database (704,705,706);

   at least a first and a second database driver (712,713,714) associated with said first and said second databases, respectively;

   said first database driver being designed to perform searches on said first database;

   said second database driver being designed to perform searches on said second database; and

   a software module (720) interacting with said first and said second drivers to achieve a desired search result.

2. The database system of claim 1 wherein said first database comprised pairs of characters separated by a predetermined association symbol.

3. The database system of claim 1 wherein said searches on said first and said second databases include logic operation on data in said databases.

4. The database system of claim 1 wherein said software module causing said first and said second database drivers to perform searches on their respective databases and combining results of said searches to achieve said desired search result.

5. A network database system including a plurality of computers coupled to a network (812), comprising:

   a first single-association database (704,705,706);

   a first driver (712,713,714) designed to perform searches on said first database;

   a second single-association database (704,705,706);

   a second database driver (712,713,714) designed to perform searches on said second database;

   said first single-association database, said second single-association database, said first database driver and said second database driver residing on different computers on said network;

   means for downloading said first single-association database, said second single-association database, said first database driver and said second database driver to a selected computer on said network, if said first single-association database, said second single-association database, said first database driver and said second database driver are not already residing on said selected computer; and

   a software module (720) on said selected computer for interacting with said first and said second drivers to achieve a desired search result.

6. A method for searching data in a database system, comprising:

   providing at least a first and a second single-association database;

   providing at least a first and a second database driver associated with said first and said second databases, respectively, said first and said second database drivers being designed to perform searches on said first and said second databases, respectively;

   providing a software module interacting with said first and said second drivers to achieve a desired search result;

   sending a first search instruction to said first database driver by said software module, said first search instruction including a first search criteria;

   searching said first single-association database by said first database driver in accordance with said first search criteria to generate a first search result;

   sending a second search instruction to said second database driver by said software module, said second search instruction including a second search criteria;

   searching said second single-association database by said second database driver in accordance with said second search criteria to generate a second search result; and

   combining said first and said second search re-

sults by said software module to obtain a desired search result.

Patentansprüch

1. Datenbanksystem mit

   - mindestens einer ersten und einer zweiten Einzelzuordnungs-Datenbank (704,705,706),
   - mindestens einem ersten und einem zweiten Datenbanktreiber (712,713,714), welche der ersten bzw. zweiten Datenbank zugeordnet ist,
   - wobei der erste Datenbanktreiber für ein Suchen in der ersten Datenbank eingerichtet ist,
   - und der zweite Datenbanktreiber für ein Suchen in der zweiten Datenbank eingerichtet ist,
   - und ein Softwaremodul (720) mit dem ersten und dem zweiten Treiber zusammenarbeitet, um das gewünschte Suchergebnis zu erreichen.

2. Datenbanksystem nach Anspruch 1, bei welchem die erste Datenbank Paare von durch ein vorbestimmtes Zuordnungssymbol getrennten Zeichen enthält.

3. Datenbanksystem nach Anspruch 1, bei welchem die Suche in der ersten und in der zweiten Datenbank logische Operationen mit den Daten in diesen Datenbänken umfaßt.

4. Datenbanksystem nach Anspruch 1, bei welchem der Softwaremodul die ersten und zweiten Datenbanktreiber veranlaßt in ihren zugehörigen Datenbanken zu suchen und die Suchergebnisse zu dem gewünschten Suchergebnis zu kombinieren.

5. Netzwerkdatenbanksystem mit einer Mehrzahl von Computern, die mit einem Netzwerk (812) gekoppelt sind, mit

   - einer ersten Einzelzuordnungs-Datenbank (704,705,706),
   - einem ersten Treiber (712,713,714), der zum Suchen in der ersten Datenbank eingerichtet ist,
   - einer zweiten Einzelzuordnungs-Datenbank (704,705,706),
   - einem zweiten Datenbanktreiber (712,713,714), der zum Suchen in der zweiten Datenbank eingerichtet ist,
   - wobei die erste und die zweite Einzelzuordnungs-Datenbank und der erste und zweite Datenbanktreiber sich in unterschiedlichen Computern des Netzwerks befinden,
   - einer Einrichtung zum Herunterladen der ersten Einzelzuordnungs-Datenbank, der zweiten Einzelzuordnungs-Datenbank, des ersten Datenbanktreibers und des zweiten Datenbanktreibers in einen ausgewählten Computer des Netzwerks, falls die erste Einzelzuordnungs-Datenbank, die zweite Einzelzuordnungs-Datenbank, der erste Datenbanktreiber und der zweite Datenbanktreiber sich nicht bereits in diesem ausgewählten Computer befinden, und
   - einem Softwaremodul (720) in dem ausgewählten Computer zum Zusammenwirken mit den ersten und zweiten Treibern, um das gewünschte Suchergebnis zu erhalten.

6. Verfahren zum Suchen von Daten in einem Datenbanksystem, umfassend:

   - Bereitstellung einer ersten und einer zweiten Einzelzuordnungs-Datenbank,
   - Bereitstellung mindestens eines ersten und eines zweiten Datenbanktreibers, welche der ersten bzw. zweiten Datenbank zugeordnet sind und welche für ein Suchen in der ersten bzw. zweiten Datenbank eingerichtet sind,
   - Bereitstellung eines Softwaremoduls, welches mit dem ersten und dem zweiten Treiber zusammenarbeitet, um das gewünschte Suchergebnis zu erreichen,
   - Senden eines ersten, ein erstes Suchkriterium enthaltenden Suchbefehls an den ersten Datenbanktreiber seitens des Softwaremoduls,
   - Suchen in der ersten Einzelzuordnungs-Datenbank seitens des ersten Datenbanktreibers entsprechend dem ersten Suchkriterium, um ein erstes Suchergebnis zu erhalten,
   - Senden eines zweiten, ein zweites Suchkriterium enthaltenden Suchbefehls an den zweiten Datenbanktreiber seitens des Softwaremoduls,
   - Suchen in der zweiten Einzelzuordnungsdatenbank seitens des Datenbanktreibers entsprechend dem zweiten Suchkriterium, um ein zweites Suchergebnis zu erhalten, und
   - Kombinieren des ersten und zweiten Suchergebnisses durch den Softwaremodul zu dem gewünschten Suchergebnis.

Revendications

1. Système de bases de données caractérisé en ce qu'il comprend au moins une première et une seconde bases de données à association unique (704,705,706), au moins un premier et un second gestionnaires de base de données (712,713,714) associés respectivement à la première et à la seconde bases de données, le premier gestionnaire de base de données étant conçu pour effectuer des recherches sur la première base de données, le se-

cond gestionnaire de base de données étant conçu pour effectuer des recherches sur la seconde base de données, et un module de logiciel (720) coopérant avec les premier et second gestionnaires pour obtenir un résultat d'une recherche désirée.

2. Système de bases de données suivant la revendication 1, caractérisé en ce que la première base de données comprend des paires de caractères séparés par un symbole d'association prédéterminé.

3. Système de bases de données suivant la revendication 1, caractérisé en ce que les première et seconde bases de données incluent une opération logique sur des données se trouvant dans ces bases de données.

4. Système de bases de données suivant la revendication 1, caractérisé en ce que le module de logiciel amène les premier et second gestionnaires de base de données à effectuer des recherches sur leurs bases de données respectives et ils combinent les résultats de ces recherches pour obtenir le résultat de la recherche désirée.

5. Système de bases de données en réseau incluant une pluralité d'ordinateurs couplés à un réseau (812) caractérisé en ce qu'il comprend une première base de données à association unique (704,705,706), un premier gestionnaire (712,713,714) conçu pour effectuer des recherches sur la première base de données, une seconde base de données à association unique (704,705,706), un second gestionnaire de base de données (712,713,714) conçu pour effectuer des recherches sur la seconde base de données, la première base de données à association unique, la seconde base de données à association unique, le premier gestionnaire de base de données et le second gestionnaire de base de données se trouvant sur différents ordinateurs branchés sur le réseau, un moyen pour décharger la première base de données à association unique, la seconde base de données à association unique, le premier gestionnaire de base de données et le second gestionnaire de base de données vers un ordinateur sélectionné sur le réseau, si la première base de données à association unique, la seconde base de données à association unique, le premier gestionnaire de base de données et le second gestionnaire de base de données ne se trouvent pas déjà résider sur l'ordinateur sélectionné, et un module de logiciel (720) sur l'ordinateur sélectionné pour coopérer avec les premier et second gestionnaires afin d'obtenir le résultat d'une recherche désirée.

6. Procédé de recherche de données dans un système de bases de données, caractérisé en ce qu'il comprend les étapes consistant à prévoir au moins des première et seconde bases de données à association unique, à prévoir au moins des premier et second gestionnaires de base de données associés respectivement aux première et seconde bases de données, ces premier et second gestionnaires de base de données étant conçus de manière à effectuer respectivement des recherches sur les première et seconde bases de données, à prévoir un module de logiciel coopérant avec les premier et second gestionnaires afin d'obtenir le résultat d'une recherche désirée, à envoyer, par le module de logiciel, une première instruction de recherche au premier gestionnaire de base de données, cette première instruction de recherche incluant un premier critère de recherche, à effectuer une recherche, par le premier gestionnaire de base de données, dans la première base de données à association unique en accord avec le premier critère de recherche, afin de produire un premier résultat de recherche, à envoyer, par le module de logiciel, une seconde instruction de recherche au second gestionnaire de base de données, cette seconde instruction de recherche incluant un second critère de recherche, à effectuer une recherche, par le second gestionnaire de base de données, dans la seconde base de données à association unique en accord avec le second critère de recherche, afin de produire un second résultat de recherche, et à combiner les premier et second résultats de recherche, par le module de logiciel, afin d'obtenir le résultat d'une recherche désirée.

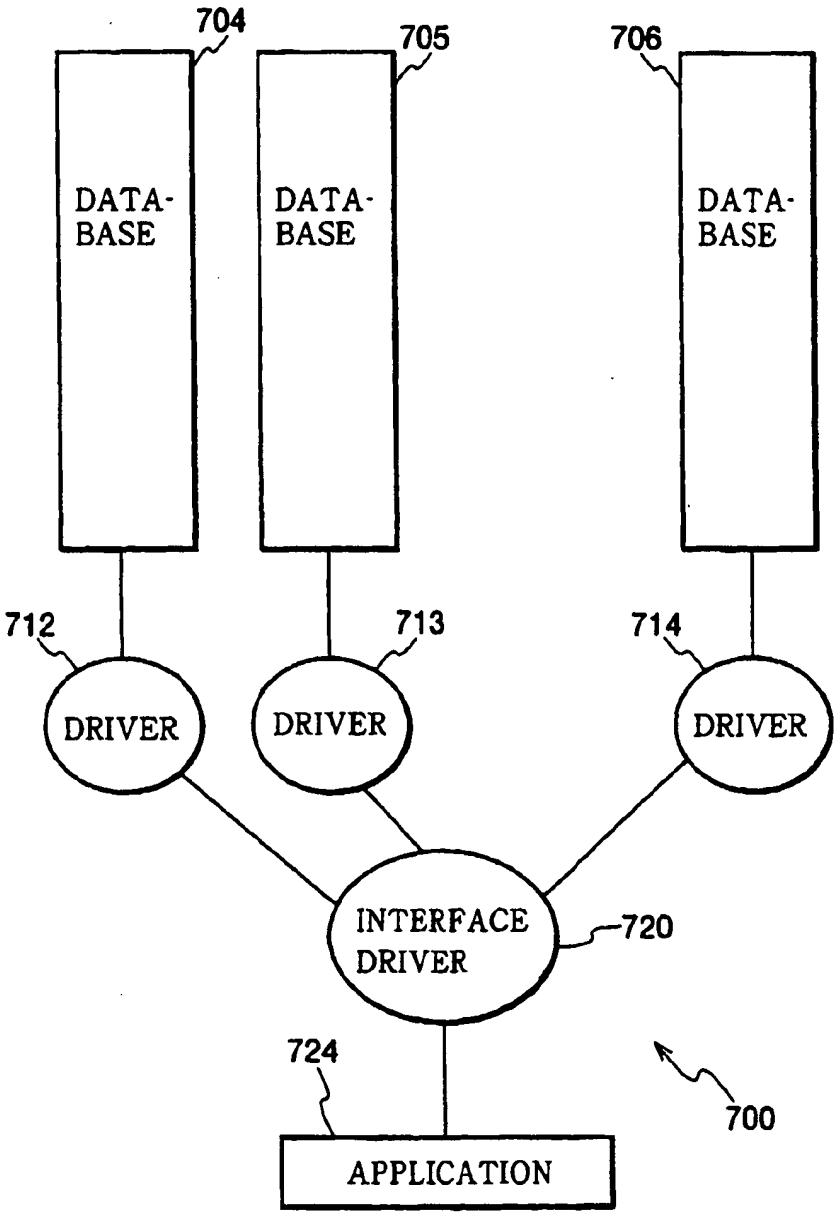
704
705
706
DATA-BASE
DATA-BASE
DATA-BASE
712
713
714
DRIVER
DRIVER
DRIVER
INTERFACE DRIVER
720
724
700
APPLICATION

FIG. 1

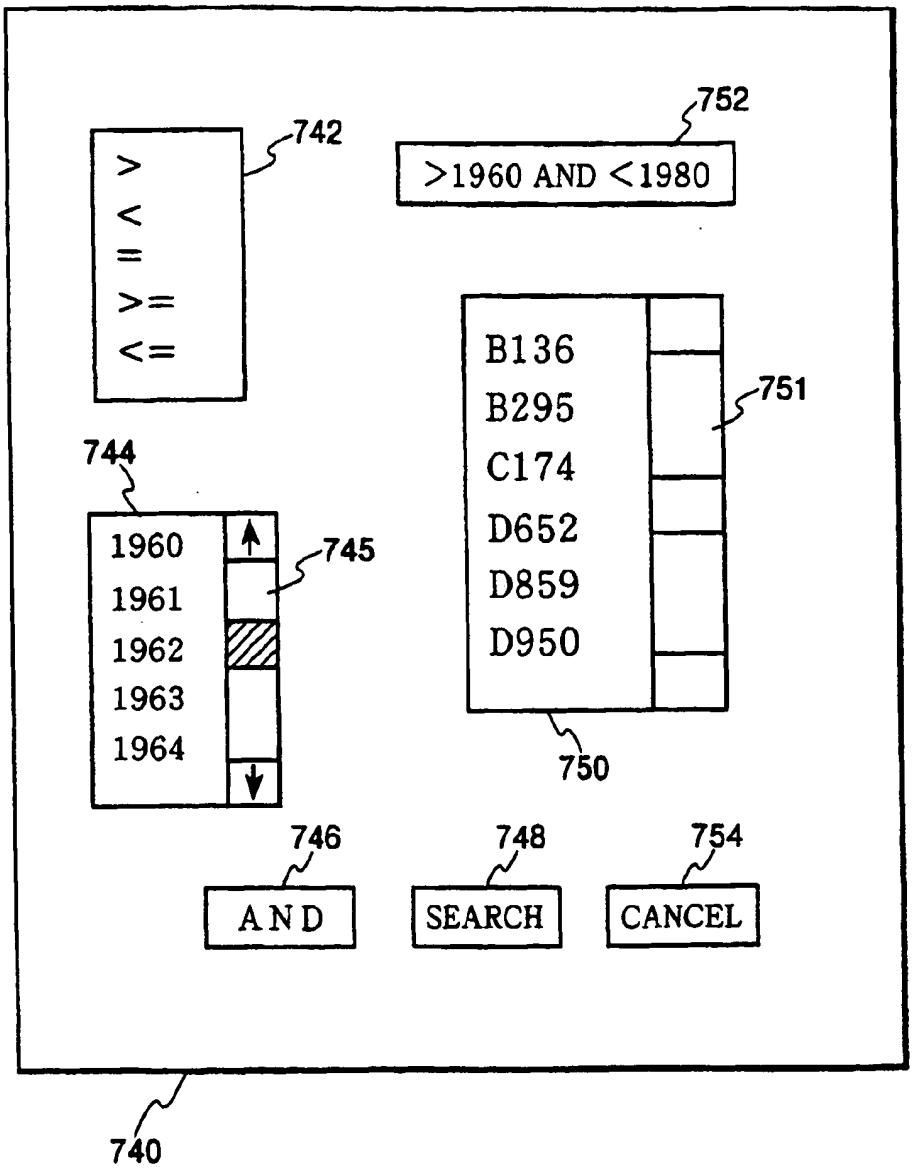
742
752
>1960 AND <1980
>
<
=
>=
<=
B136
B295
C174
D652
D859
D950
751
744
1960
1961
1962
1963
1964
745
750
746
748
754
AND
SEARCH
CANCEL
740

FIG. 2

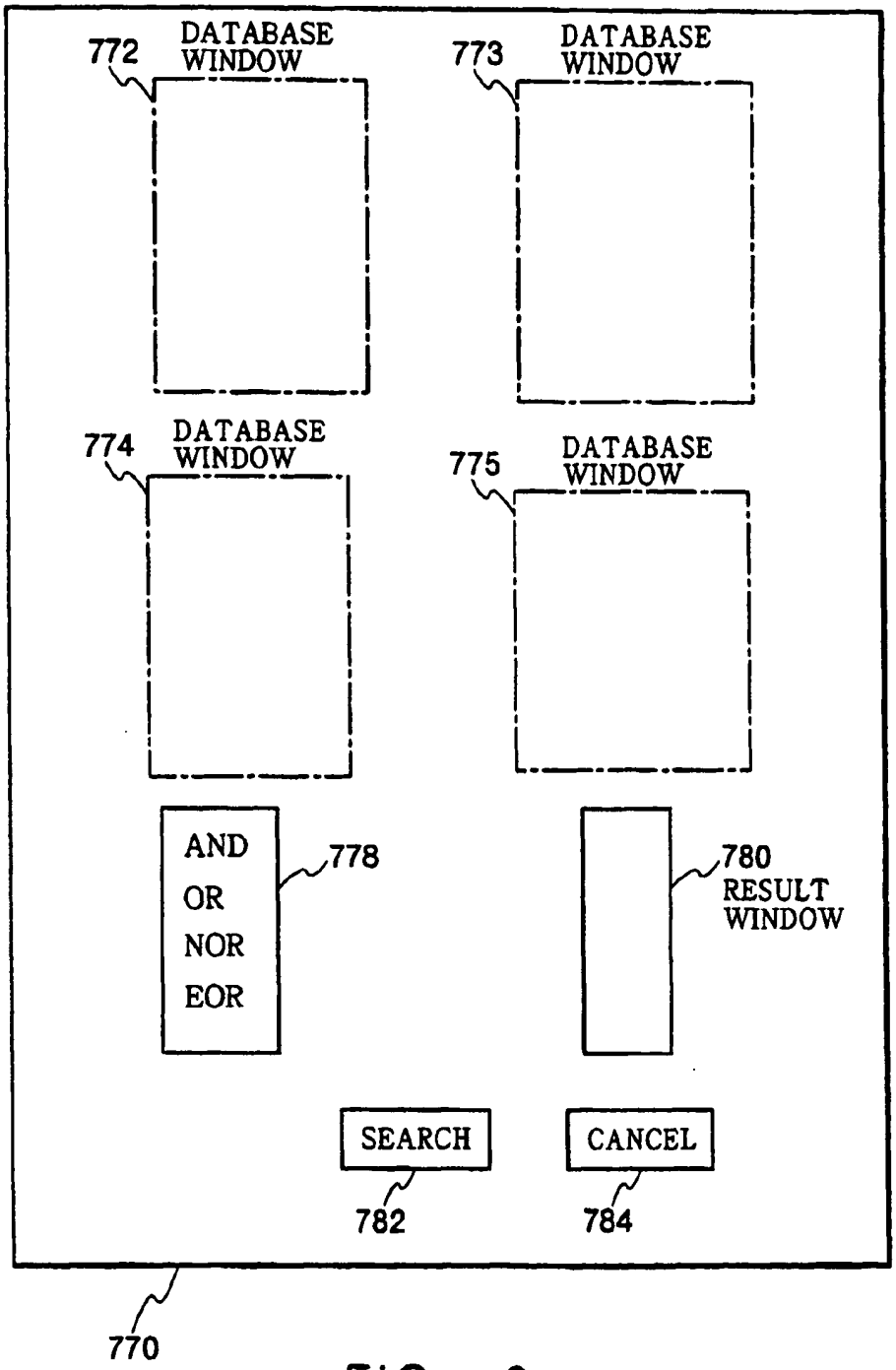
772
DATABASE WINDOW
773
DATABASE WINDOW
774
DATABASE WINDOW
775
DATABASE WINDOW
AND
OR
NOR
EOR
778
780
RESULT WINDOW
SEARCH
CANCEL
782
784
770

FIG. 3

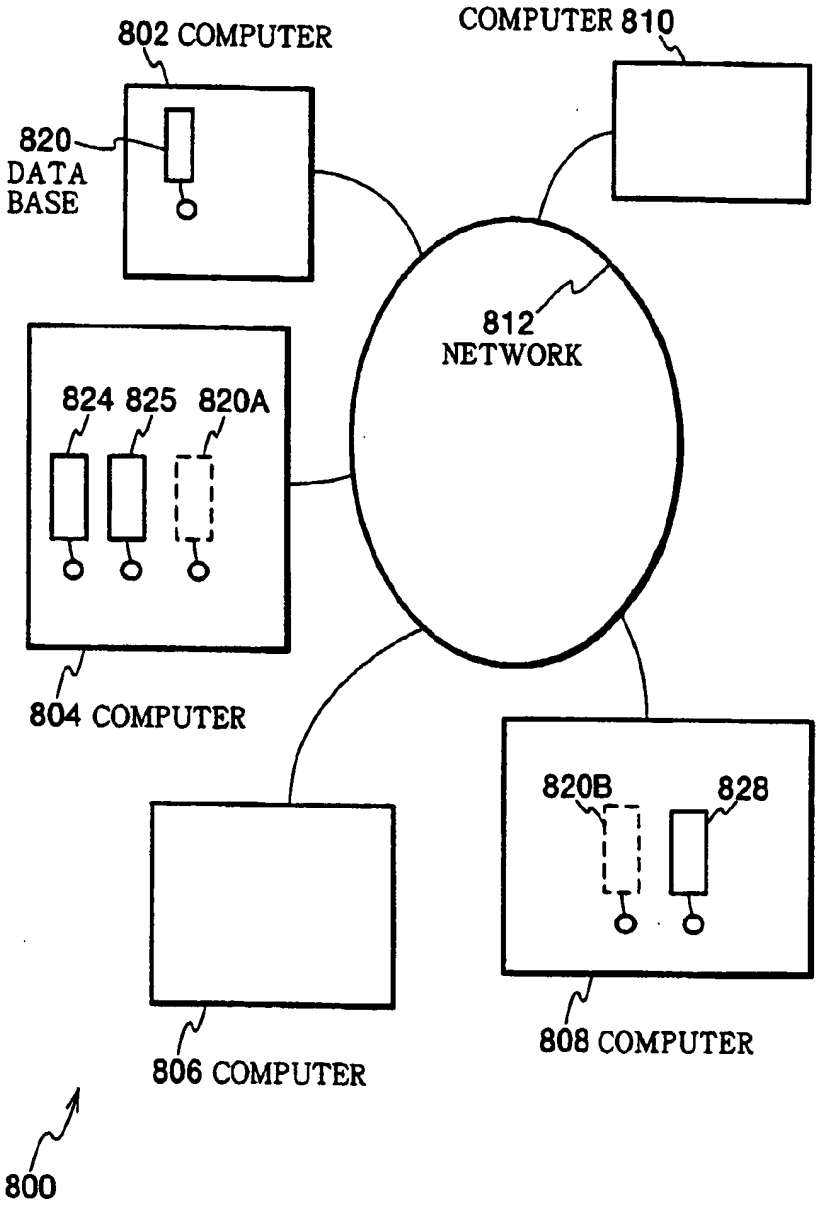
802 COMPUTER
COMPUTER 810
820
DATA
BASE
812
NETWORK
824 825 820A
804 COMPUTER
820B
828
806 COMPUTER
808 COMPUTER
800

FIG. 4

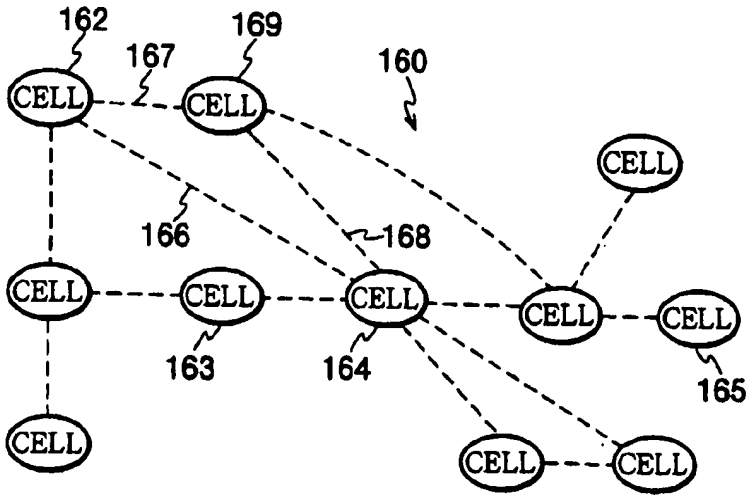
162
169
167
160
CELL
CELL
CELL
166
168
CELL
CELL
CELL
CELL
CELL
163
164
165
CELL
CELL
CELL

FIG. 5

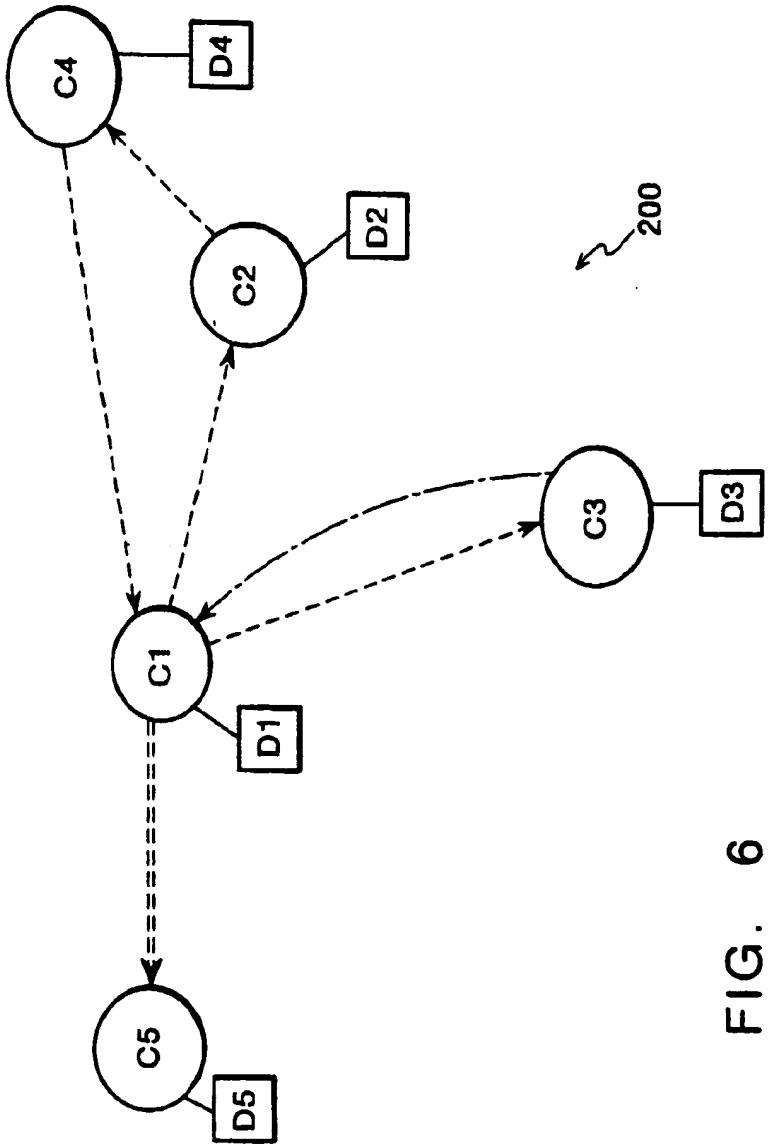

FIG. 6

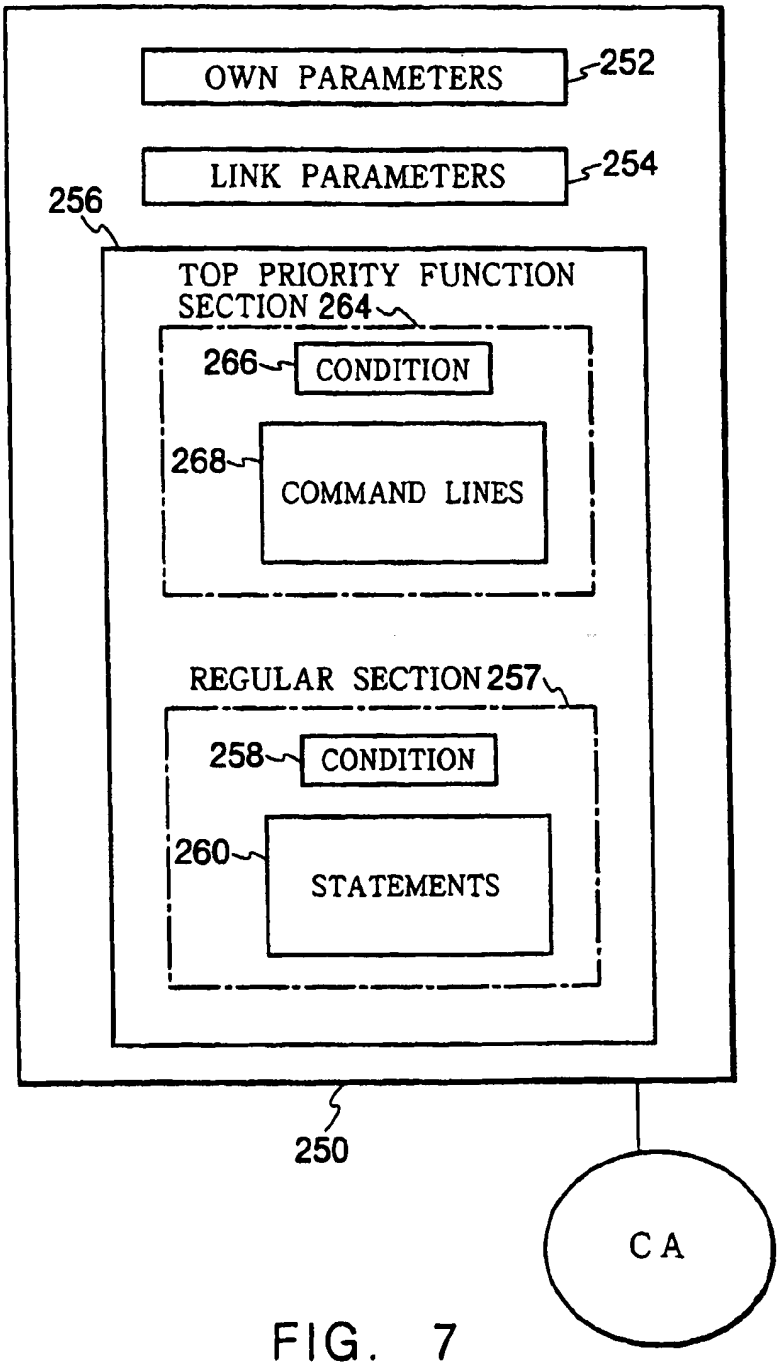
OWN PARAMETERS
252
LINK PARAMETERS
254
256
TOP PRIORITY FUNCTION SECTION 264
266
CONDITION
268
COMMAND LINES
REGULAR SECTION 257
258
CONDITION
260
STATEMENTS
250
CA

FIG. 7

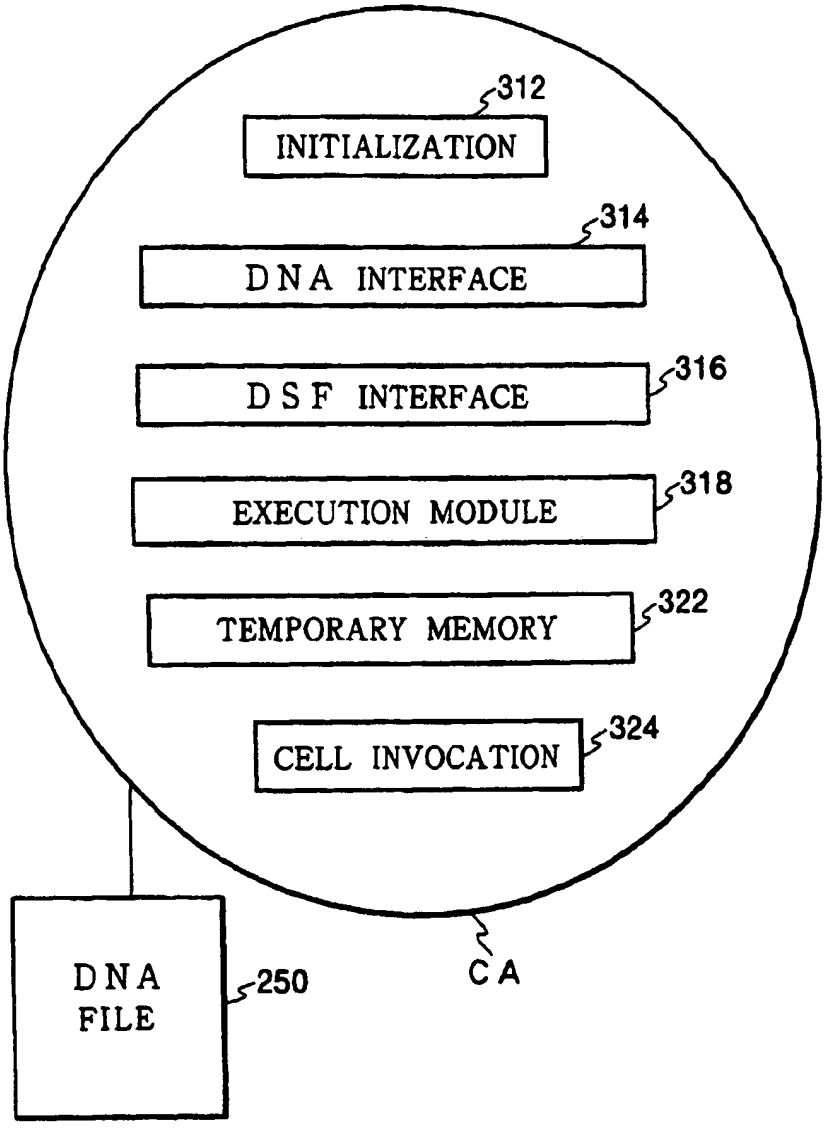
312
INITIALIZATION
314
DNA INTERFACE
316
DSF INTERFACE
318
EXECUTION MODULE
322
TEMPORARY MEMORY
324
CELL INVOCATION
CA
DNA FILE
250

FIG. 8

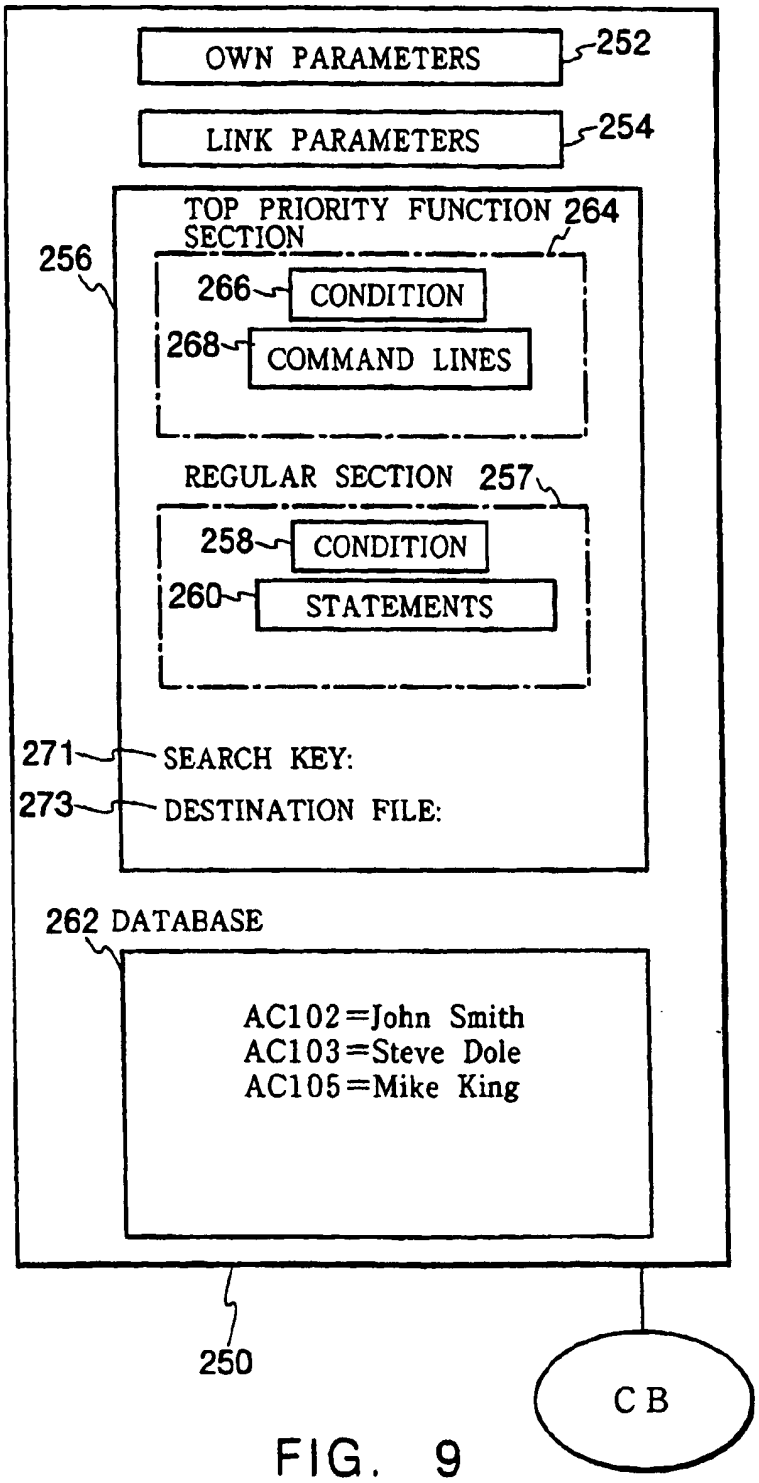
OWN PARAMETERS
252
LINK PARAMETERS
254
TOP PRIORITY FUNCTION SECTION
264
256
266
CONDITION
268
COMMAND LINES
REGULAR SECTION
257
258
CONDITION
260
STATEMENTS
271
SEARCH KEY:
273
DESTINATION FILE:
262 DATABASE
AC102=John Smith
AC103=Steve Dole
AC105=Mike King
250
CB

FIG. 9

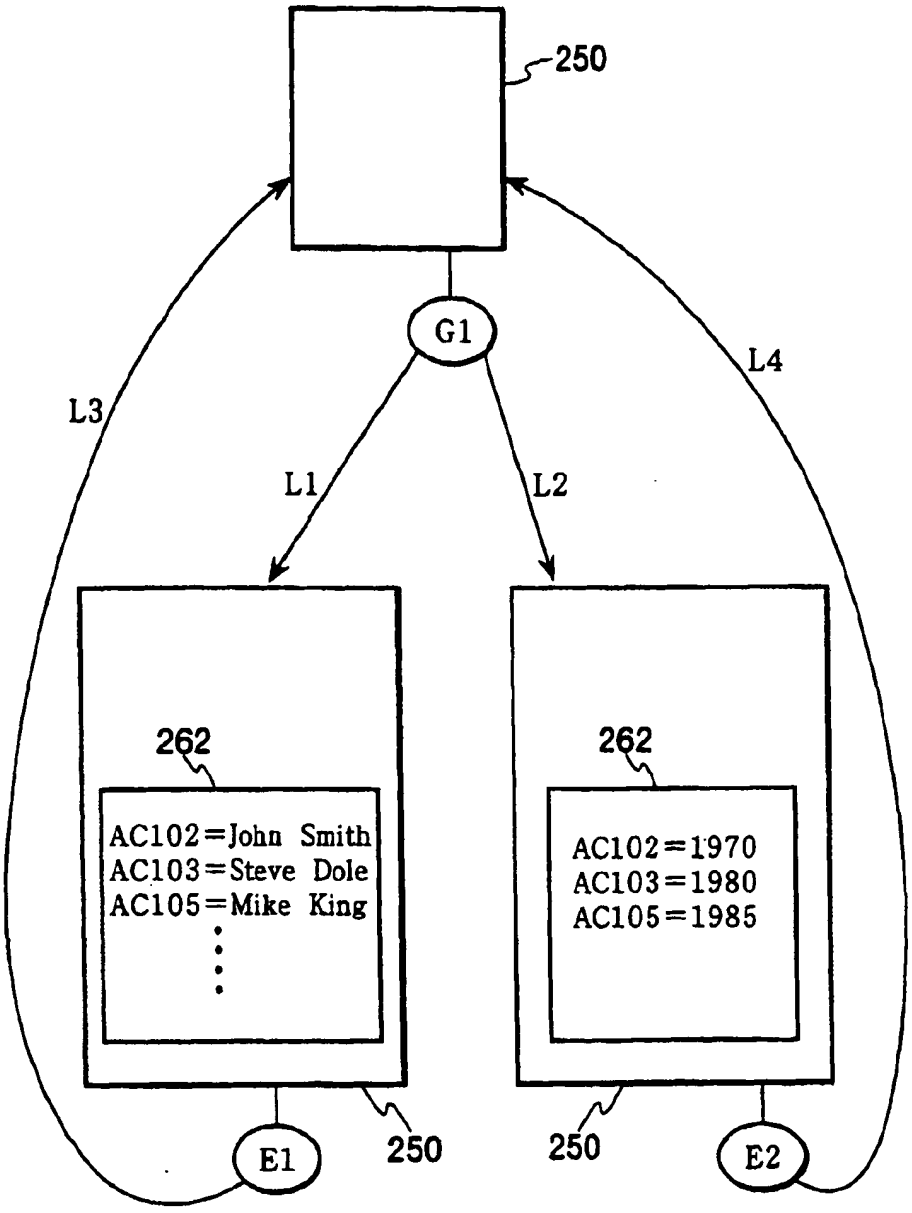
250
G1
L3
L4
L1
L2
262
AC102=John Smith
AC103=Steve Dole
AC105=Mike King
262
AC102=1970
AC103=1980
AC105=1985
E1
250
250
E2

FIG. 10

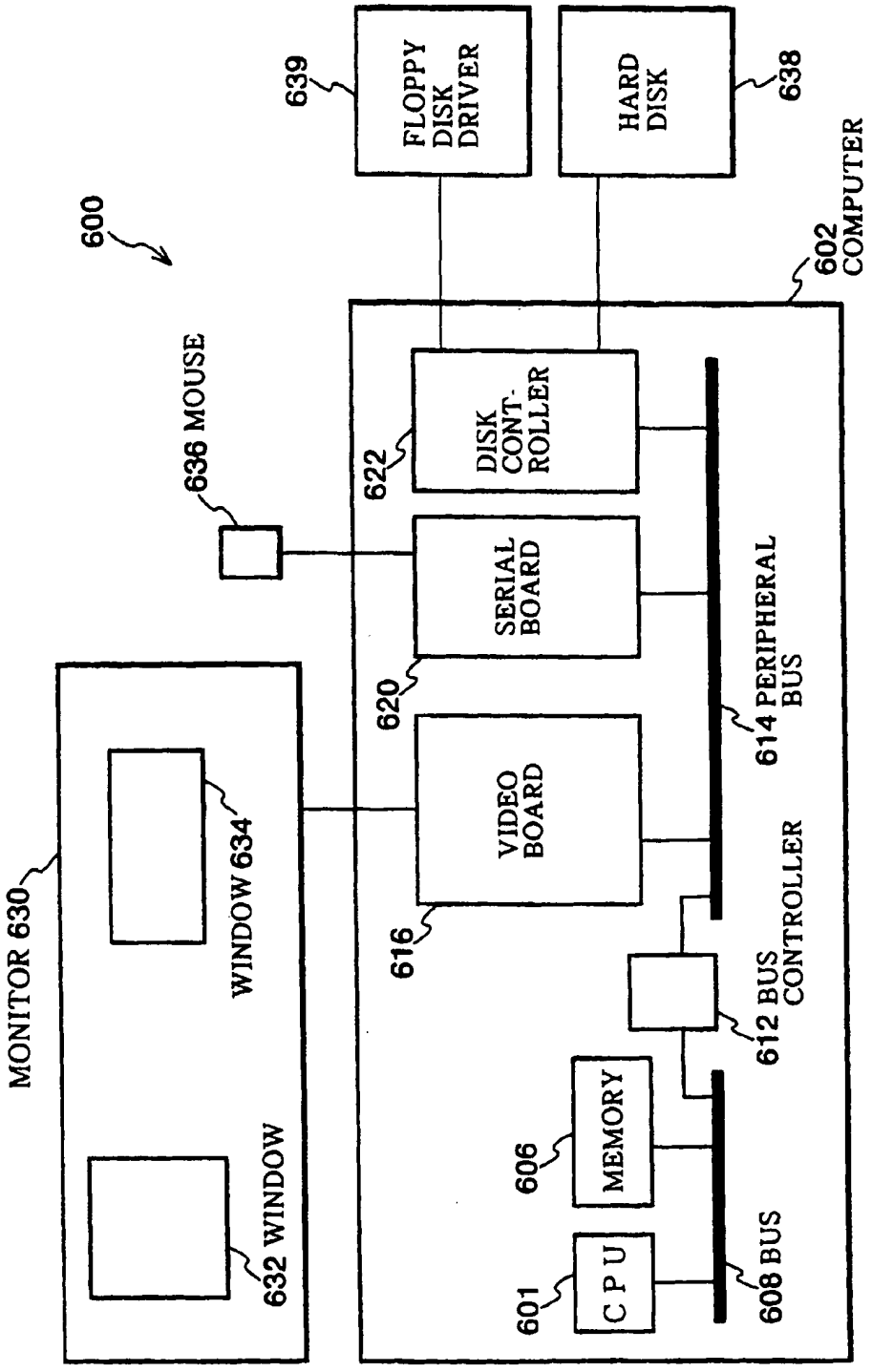
600
639
FLOPPY DISK DRIVER
HARD DISK
638
602 COMPUTER
636 MOUSE
622
DISK CONT-ROLLER
SERIAL BOARD
620
614 PERIPHERAL BUS
VIDEO BOARD
616
MONITOR 630
WINDOW 634
612 BUS CONTROLLER
606
MEMORY
601
C P U
608 BUS
632 WINDOW

FIG. 11